import express from "express";

import helmet from "helmet";

import rateLimit from "express-rate-limit";

import bcrypt from "bcrypt";

import jwt from "jsonwebtoken";

import sanitizeHtml from "sanitize-html";

import { Pool } from "pg";

const app = express();

const db = new Pool({ connectionString: process.env.DATABASE\_URL });

app.use(helmet()); // secure headers

app.use(express.json({ limit: "5kb" })); // limit input size

app.use(rateLimit({ windowMs: 60000, max: 60 })); // rate limiting

// signup

app.post("/signup", async (req, res) => {

const hashed = await bcrypt.hash(req.body.password, 12);

await db.query("INSERT INTO users(email, password\_hash) VALUES($1,$2)", [

req.body.email,

hashed,

]);

res.send("User created");

});

// login

app.post("/login", async (req, res) => {

const { rows } = await db.query("SELECT id,password\_hash FROM users WHERE email=$1", [req.body.email]);

if (!rows.length || !(await bcrypt.compare(req.body.password, rows[0].password\_hash)))

return res.status(401).send("Invalid credentials");

const token = jwt.sign({ sub: rows[0].id }, process.env.JWT\_SECRET, { expiresIn: "1d" });

res.cookie("auth", token, { httpOnly: true, secure: true, sameSite: "lax" }).send("Logged in");

});

// auth middleware

function auth(req, res, next) {

try {

req.user = jwt.verify(req.cookies.auth, process.env.JWT\_SECRET).sub;

next();

} catch { res.status(401).send("Unauthorized"); }

}

// create post (with sanitization)

app.post("/posts", auth, async (req, res) => {

const safe = sanitizeHtml(req.body.content, { allowedTags: ["p", "b", "i"] });

await db.query("INSERT INTO posts(author\_id,title,content) VALUES($1,$2,$3)", [

req.user, req.body.title, safe,

]);

res.send("Post created");

});

export default app;